ESA On-board Software Systems Section

Maria Hernek
Marek Prochazka

Requirements Modelling:
A Verification Method for ISVV
Outline

- Background
- Purpose of activity
- Activity roadmap
- Activity execution
  - Modelling
  - Model checking
  - Contribution to validation
- Conclusions, product findings and process improvements
Background

- ESA has initiated a number of activities for ISVV method assessment and process improvement for cost and effort efficiency

- ESA ISVV Guide, Issue 2, contains improvements in all method areas

- This presentation is limited to requirements verification and validation with focus on UML/SysML modelling
Goal

- Assess requirements modelling and model checking as methods for requirements verification

That is,

- Take requirement specification written in plain English
- Create a corresponding model
- Use various model checking tools to verify the specification
- Contribution to Validation: Definition of test cases
Activity Roadmap

Written Requirements → Modelling → Model

Model Checking → Contribution to Validation
Which “Model Checking”?  

- **Formal: Maude**  
  - model execution, proving properties  

- **Semi-formal: UML Object Constraint Language (OCL)**  
  - Magic Draw (SysML)  

- **Informal: Checklists for dependability**  
  - Using the SysML model
Reference Model in UML
UML Class Diagram

- Classes represent concepts in the system and their associations
- Class diagram provides the system overview

Findings:
- Internal inconsistencies (same aspects modelled differently in different diagrams)
- Completeness: Unspecified multiplicities
UML Activity Diagram

- Defines activities and their flow control
- Concurrent activities

Findings:
- Concurrent execution
  - Related activities not being specified
- Error conditions and FDIR
  - Related error conditions not specified
  - Missing requirements defining alternative behaviours
UML State Machine Diagram

- Used to express states of the system and corresponding state transformations

- Findings:
  - Pairs of states not been taken into account
  - Transitions not being specified
  - Missing guards on transitions
UML Requirements Modelling: Overall Assessment

- **UML is easy to use**
  - It is flexible in having many different diagram types
  - It is flexible in the use of the diagrams

- **Difficult to find a “good” UML representation**
  - UML is a graphical tool, not a method

- **Difficult to find/select correct abstraction level**
  - Due to flexibility (e.g. messages in sequence diagrams)

- **Flexibility of UML allows imprecise and informal models**
  - E.g. messages in sequence diagrams
  - But formal models are difficult to comprehend…
UML Model Checking (I)

- Mapping between formal semantics of UML and model checking language

- Determine relevant types of checks
  - Model checking can be used to verify that all possible runs of a model satisfies the properties
  - Especially useful for multithreaded systems

- Model checking language – specialist needed
UML Model Checking (II)

- Formalising requirements/models removes ambiguity and helps requirements review
- Executable models increase understanding
- UML diagram does not contain enough information to create formal model, assumptions have to be introduced
- UML diagram has to be constrained to allow model checking translation
- When complexity of model increases the number of possible states increases exponentially – state explosion
- Abstraction can be used but keep in mind that you verify a simplified model
- Findings were done as part of the formalisation, no findings when executing model – all findings related to completeness
  - Could be that the use case was too simple (Data Handling)
Maude: An Example

rl [A-acquire-lock] :

< “Object A” : Object | needResource : “Sensor” >
< “Sensor” : SharedResource | isLockedBy : null >
=>

< “Object A” : Object | needResource : “Sensor” >
< “Sensor” : SharedResource | isLockedBy : “Object A” >.

- The part before “=>” specifies what must be the state of the model for the rule to be applicable
- The part after “=>” specifies how the system transforms if the rule is applied
UML Model Checking: Summary

- UML tool: Enterprise Architect (Sparx Systems)
- Approx. 2-3 hrs per requirement for modelling!

- 34 findings
  - 24 finding during modelling
  - 10 during model checking or pre-modelling reading
  - Findings related to missing or incomplete requirements

- Model checker: Maude

- Manual translation to Maude
  - Translation from UML to model checker language not supported:
Modelling in SysML
Requirements Modelling in SysML (I)

- Add requirements identifier to the model
  - Storing text in a standardized format
  - Relate requirements between themselves
    - Associations between requirements
    - Associating the same requirement with several elements
    - Visualize requirements
    - Maintaining and tracking of requirements and their relationships

- Model system context (not a predefined diagram)
  - Represent direct environment
  - Initial information about communication flow to and from system
  - Further detailed level, incoming and outgoing information and identifying and analysing actor by actor
Model for System Context
System Context with Information Flows

[Diagram of system context with information flows]
Requirements Modelling in SysML (II)

- **Advantages**
  - Integrates requirements

- **Difficulties**
  - System context definition
  - Correct adjustment of details required
  - When are requirements fully realized?

- **Possibly improvements**
  - Replace use cases with high-level activity diagram
SysML Modelling: Summary

- SysML tool MagicDraw from No Magic Inc.
- 250 requirements
  - 163 diagrams (block, activity, use cases etc)
  - Approx. 2hrs per requirement!
  - Approx. 60 findings during the modelling process
    - 50 completeness
    - 10 consistency
SysML Model Verification

- Model checking based on OCL scripts
  - Predefined suits for correctness
  - Predefined suits for completeness
  - Additional rules

- Manual checking based on checklist for dependability

- Findings
  - Few from predefined OCL suites
  - More from customized OCL suites
  - In total the number of findings was small comparing to the modelling phase
  - No finding from checklist
## Examples of OCL Rules

<table>
<thead>
<tr>
<th>CORR-01</th>
<th>Max_composite_multiplicity_1</th>
<th>error</th>
<th>Max multiplicity on composed end should be 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORR-02</td>
<td>no_method_for_abstract_operation</td>
<td>error</td>
<td>Abstract operation can not have method defined</td>
</tr>
<tr>
<td>CORR-03</td>
<td>operation_and_classifier_abstract</td>
<td>error</td>
<td>If at least one operation of classifier is abstract, classifier should be abstract</td>
</tr>
<tr>
<td>CORR-04</td>
<td>datatype_operation_is_query</td>
<td>error</td>
<td>Operations of datatype must be queries</td>
</tr>
<tr>
<td>CORR-05</td>
<td>Leaf_not_abstract</td>
<td>error</td>
<td>Leaf Element should not be abstract</td>
</tr>
<tr>
<td>CORR-06</td>
<td>no_inheritance_from_leaf</td>
<td>error</td>
<td>Can not inherit additional classes from a leaf element</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CMP-07</th>
<th>States_incoming</th>
<th>warning</th>
<th>State is without incoming transitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMP-08</td>
<td>Name_for_actor</td>
<td>error</td>
<td>Actors should be named</td>
</tr>
<tr>
<td>CMP-09</td>
<td>operation_for_calimessage</td>
<td>error</td>
<td>Call message should have operation assigned</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SML-05</th>
<th>Viewpoint3</th>
<th>warning</th>
<th>The property ownedAttributes must be empty</th>
</tr>
</thead>
<tbody>
<tr>
<td>SML-06</td>
<td>Block1</td>
<td>warning</td>
<td>Property metaclass that is typed by a block and is owned by an Association may not have a name and may not be defined as a navigable owned end of the association.</td>
</tr>
<tr>
<td>SML-07</td>
<td>BlockProperty1</td>
<td>warning</td>
<td>The type of the block property must be a block</td>
</tr>
</tbody>
</table>
Contribution to Validation

- How can modelling and/or model verification contribute to validation
  - i.e. how can this method for requirements verification support identification of validation test cases

- Test cases identification
  - Based on the missing requirements
  - Generating test case and comparing with the test cases identified by the software supplier
  - Tedious work
Conclusions (I)

- Requirements verification through modelling forces verifier to understand the system

- Depending on modelling approach, do we get the same or different findings?

- Most findings identified during the modelling phase

- Model checking revealed few findings only, often already found earlier
  - Possibly because of already performed verification activity
Conclusions (II)

- Modelling promoted as requirements verification method

- Model checking found not mature yet

- Contribution to validation still to be assessed properly